Resilience of Structured Peer to Peer Systems: Analysis and Enhancement

Dong Xuan, Sriram Chellappan and Xun Wang

Abstract

In this paper, we attempt to provide an extensive and detailed summary of existing work in the area of Structured Peer-to-Peer (P2P) systems resilience. Broadly speaking, resilience of a P2P system is the ability of the system to defend against threats due to system dynamics (in terms of node joins and leaves) and malicious nodes. The coverage in this paper includes resilience metrics, threats, approaches to analyze resilience and enhancements proposed to individual system design features to enhance resilience. Specifically, we first discuss important design features of popular structured P2P systems, the performance metrics and several possible threats and attack scenarios that threaten the performance of structured P2P systems. We then provide a broad overview of approaches to analyze resilience employed by researchers in this area. Following this, we describe an analytical approach developed by us to analyze resilience of structured P2P systems. Our approach is Markov-Chain based and can be applied to analyze systems with relatively stable size and uniformly distributed nodes. We then discuss the insights we obtain on resilience using our analytical approach. Following in the same direction, we then discuss enhancements to individual structured P2P system design features that have been proposed by researchers to increase overall system resilience. We then discuss two of our enhancements, namely CAN-SW (CAN Small World) and RChord (Reverse Chord) to enhance resilience of the CAN and Chord system respectively. Our approaches to design these enhancements derive inspiration from the importance of redundancy and the richness in graph-theory.

Index Terms

Structured Peer to Peer Systems, Resilience

Dong Xuan, Sriram Chellappan and Xun Wang are with the Department of Computer Science and Engineering, The Ohio-State University, Columbus, OH 43210. E-mail: {xuan, chellapp, wangxu}@cse.ohio-state.edu.

This work was partially sponsored by NSF under contract number ACI-0329155. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

I. Introduction

Peer-to-Peer (P2P) systems are a new paradigm of communication systems in which all computers are treated as equals or peers in the network. Individual computers may share hard drives, CD-ROM drives, other storage devices, files etc. with the other computers on the network. This is different from a client/server set-up in which most of the computers (clients) tend to share resources from one main computer (the server). P2P systems have several advantages as follows and more. The system model is inherently distributed. There is no need for huge servers and enormous indexing. Single point of failures are eliminated due to the absence of dependencies on any specific server. Sharing of available content does not require a publication step like creating a web page or uploading to a server. Anonymizing the communication between the requester and provider is easier. The systems can be implemented with localized security considerations and it is flexible to modify them.

Broadly speaking, there are two classes of P2P systems, namely unstructured and structured. Unstructured systems like Gnutella [1], Kazaa [2] and Freenet [3] are constructed on the fly, without any regularization on the connectivity among peers or the searching mechanism. Such networks are easier to build and so is the maintenance (or the lack of it). Typically, new nodes randomly connect to existing alive nodes in the network and the searching process for resources is flooding (called as Breadth First Search) or iterative searching (called as Depth First search).

Another class of P2P systems is structured P2P systems [4], [5], [6], [7], where the system (comprising of peers) follows a predetermined structure. This structure needs to be maintained by participating peer nodes for correctness. Structured P2P systems typically use Distributed Hash Table functionality to assign identifiers to nodes and the resources. Structured P2P systems enjoy efficient look-up mechanisms that increase data availability and bound the time/ messages required to find available data.

It is envisaged that eventually millions of users will demand access to these systems. In such a situation, it will be prohibitive to validate or constraint membership to such networks. This raises serious questions about the trustworthiness of nodes that join these networks along with the dynamics associated with the

membership (in terms of node joins and leaves). The P2P system should be able to operate in the presence of malicious nodes and dynamics. That is, the system needs to be Resilient. Resilience of P2P systems can be thought of as being impacted in two ways. The first is the resilience of P2P systems under node failures. The second is the resilience of the P2P systems to attacks by malicious nodes.

In this paper, we will focus our attention on resilience of structured P2P systems under the presence of node dynamics and attacks. We will first provide a brief introduction to the design features and performance metrics of structured P2P systems followed by threats associated with structured P2P systems. We then describe approaches to analyze the resilience of structured P2P systems followed by enhancements to the design features of these systems to make them more resilient. Our discussions on the analysis and enhancements will first provide with a survey of related work followed by our approaches to do the same.

II. Structured P2P Systems and Their Threats

A. Structured P2P systems

Structured P2P systems, as the name suggests contains peer nodes arranged in a fixed structure. Structured P2P systems use Distributed Hash Tables (DHT) for resource management and searching. Typically DHT's associate hash values (keys) with content or resources present in the system. Participants in the system each store a small section of the contents of the hash table, improving load balancing and searching. Secure hashes such as MD5 [8] and SHA1 [9] are commonly used to assign collision free identifiers to nodes and resources. What follows is a very brief background of four popular structured P2P systems. For more details on these systems, interested readers can read the corresponding literature.

a) CAN: CAN [5] resembles a hypercube geometry. CAN uses a d-torus that is partitioned amongst nodes such that every node owns a distinct zone within the space. As explained in CAN [5], a CAN node's identifier is a binary string representing its position in the space. For an N node system, when $d = logN$ [1] dimensions, the neighbor sets in CAN are exactly those of a $logN$- dimensional hypercube. Each node has $logN$ neighbors; neighbor i differs from the given node on only the i^{th} bit. The distance between

[1]Unless otherwise specified, $\log N$ denotes $log_2 N$.

two nodes is the number of bits on which their identifiers differ and routing works by greedy forwarding to reduce this distance. Thus routing is effectively achieved by correcting bits on which forwarding node differs from the destination. The path length in CAN scales as $O(\frac{d}{4}n^{\frac{1}{d}})$.

b) Chord: Chord [4] is a distributed hash table storing key-value pairs. Keys are hashed into a circular N-bit identifier space $[0; 2^N)$ (identifiers on the circle are imagined as being arranged in increasing order clockwise). Each node is assigned a unique identifier (id) drawn uniformly at random from this space, and the distance from a node m to a node k is the clockwise distance on the circle from m's id to k's id. Each node m maintains a link to the closest node k that is at least distance 2^i away, for each $0 \leq i < N$. The set of nodes forming a Chord network is known as a Chord ring. In Chord, a node can route to an arbitrary destination in $logN$ hops because the routing semantics in Chord at each hop cuts the distance to the destination by half.

c) Pastry: In Pastry [6], nodes are responsible for keys that are the closest numerically (with the keyspace considered as a circle similar to Chord). The neighbors consist of a $LeafSet$, L, which is the set of L closest nodes (half larger, half smaller) for correctness in routing. To achieve more efficiency, Pastry has another set of neighbors spread in the key space. Routing consists of forwarding the query to the neighboring node that has the longest shared prefix with key. In the event of node failures, Pastry uses a repairment algorithm discussed in [6].

d) Tapestry : Tapestry [7] is an overlay location and routing infrastructure where the nodes maintain the routing table called as $neighbormaps$. It routes along the nodes following the longest prefix routing similar to the one in the CIDR IP address allocation architecture. The Tapestry location mechanism is almost similar to the Plaxton location scheme [10] but has more semantic flexibility.

A host of other structured P2P systems have been designed. The Viceroy [11] algorithm emulates the operation of a traditional Butterfly network. A constant routing state is maintained at each node and routing is achieved in three stages; each in $O(logN)$ steps towards the destination. The PRR [10] scheme employs a tree based structure. Routing works greedily towards the destination by successively correcting

the highest order bits on which the forwarding node differs from the destination. The Kademlia [12] system uses the numerical value of the Exclusive OR (XOR) as the routing distance between two nodes and routing is through greedy forwarding.

The performance of any P2P system can be evaluated based on three metrics. They are (1) Data Availability, (2) Resource Consumption and (3) Messaging Overhead. Data availability refers to the ability of the system to locate the requested data correctly. One choice to quantify it is the Hit Ratio defined as the ratio of the number of successful queries to the total number of queries generated. Resource consumption during routing is estimated by measuring Hop Length and Latency. Hop length refers to the number of peer to peer hops (not the number of hops at the network layer) taken by the system to locate a resource. Most designs employing DHT functionality can locate available data within $O(logN)$ hops for an N node system. The Latency from the request to the response is the actual delay in time seen by the user. Systems try to route messages based on proximity estimates to other peers in order to minimize the latency.

The third metric is the messaging overhead associated with system repair due to the inherent dynamics associated with the nodes in terms of joining and leaving the network frequently. The system has to update pointers that otherwise incorrectly point to lost or new data, update routing tables reflecting the new changes, correctly route current queries. Structured P2P systems have to do more messaging in this realm in order to maintain the structure, where as unstructured P2P systems do not handle dynamics with the intention of repairing the system.

B. Threats

This section provides an overview of the threats associated with P2P systems. To clarify the reader, a threat when implemented or executed by a node (typically malicious) becomes an attack. Threats can be classified into two types Passive and Active. Passive threats occur due to inherent threats in the system and not due to malicious participants while Active threats involve malicious nodes deliberately compromising the system.

Passive threats in P2P systems are typically due to dynamics caused by frequent node failures and joins.

- Node failures: If preplanned node failures occur, content can be distributed across other nodes. But if failures occur unexpectedly, existing content may be un-obtainable until the node restarts or some replication mechanisms are in place.
- Node joins: Although node joins are useful in the sense that content can be made more available and more distributed effectively, it comes at a cost of maintenance overhead that should be minimized. This overhead depends on the number of nodes, the geometry of the system and the maintenance protocol.

On the other hand, Active threats are due to the presence of malicious nodes. Such threats can be further classified as follows.

- Routing Attacks: Malicious nodes present in the system can disrupt operations by not obeying the routing logic. They can route to nodes farther than the destination with the intention of increasing latencies or path length; they can drop queries compromising data availabilities; they may advertise false routing table information to other nodes.
- Sybil Attacks: Malicious nodes may indulge in Sybil attacks [13] where a node masquerades itself with identities of other nodes. If node identities are not certified, it is very easy for malicious nodes to execute Sybil attacks. If some popular content is replicated, then the redundancy in the system can be seriously compromised under Sybil attacks.
- Freeloading: In the realm of file sharing systems several works like [14], have shown the problem of freeloading. A user (or node) is a P2P system is a freeloader, when it obtains content from other users, but refuses to share its content to others. This results is significant imbalance in long term operation and defeats the basic tenet of all users (or nodes) being peers.
- Miscellaneous Attacks: There are several other types of attacks possible. Malicious nodes can launch Distributed Denial of Service (DDoS) attacks by bombarding a peer with queries. The intention is to prevent that peer from providing service to other benign peers in the system. A malicious node also can declare that a particular node X is the holder of a resource, when in fact node X may not

posses the resource. In other words, trusting one node to speak out for another node can be taken advantage of by malicious nodes.

All of the above attacks can be orchestrated by one or more nodes separately. However when individual attackers coordinate to conduct these attacks, the impacts can be severe. Typical scenarios of coordinated attacks include the following; since all structured P2P systems follow a rigid structure, if the malicious nodes locate themselves at strategic locations, performance can be significantly compromised; When a new node joins a network, it can inadvertently contact a malicious node first. If a set of malicious nodes form a parallel network among themselves, then this newly joined node can neither contribute nor enjoy resources.

III. Analyzing Resilience of Structured P2P Systems

A. A Survey

In broad terms, the resilience of a P2P system is its ability to maintain performance levels under the presence of malicious nodes and dynamics. The better the performance is, the more resilient is the system. In [15], a term called Static Resilience is defined as the ability of the P2P system to locate resources in the period of transition between node failures and full recovery by the system. The importance of Static Resilience is a measure of how quickly the recovery algorithm has to work. Systems with low Static resilience need to have faster recovery algorithms rather that those with higher levels of static resilience. Although Static Resilience is an important metric, it does not quantify the resilience in the presence of malicious nodes that other works tries to address.

Typically there are three approaches to analyze P2P resilience. Some works focus on a general description of threats and attacks, their impacts on system resilience and optimizations to the system features. Such works like [16], [15] use representative case studies and/ or simulations to study resilience. Another class of works employs Graph-theoretic approaches to study resilience [15], [27], [18], [19]. All structured P2P systems can be modeled as graphs and the richness of graph theory is employed to dissect the system's

behavior and study its resilience. A third approach uses analytical methods to study resilience. Such works like [17] mathematically model the P2P system behavior to obtain insights into its performance.

An introductory analysis on the impact of geometry on resilience of P2P systems on Resilience and Proximity is presented in [15]. The authors dissect popular P2P systems in terms of their geometry, flexibility in neighbor and route selection, and the actual routing algorithms used to locate resources. Based on formal arguments and simulations, the authors highlight the importance of flexibility in Route selection, the presence of sequential neighbors (neighbors to which one can route and be sure of making progress towards all destinations), building routing tables based on proximity to enhance overall system resilience. In the same vein, Sit and Morris [16] analyze security issues in p2p systems by specifying several possible attacks scenarios and countermeasures.

Graph Theoretic approaches to analyze resilience are presented in [18], [19]. The P2P systems are modeled as graphs and certain graph-theoretic properties like connectivity, diameter are analyzed. Based on findings, the authors propose P2P systems based on classes of graphs that do possess desirable properties. Some instances of such graphs are de Bruijn graphs [20], [18] and random graphs [19].

B. A Markov-chain based Analytical Approach

In this section, we will first describe our general approach to analyze resilience of structured P2P systems. Following that, using a case study, we will discuss how to apply this approach to P2P systems under node failures.

As we know in P2P systems during the data look-up process, a node forwards a query to the next node based on its current routing table. The routing process can be modeled as a discrete absorbing *Markov chain*:

- Define a stochastic process $\{X_h : h = 0, 1, \cdots\}$, where random variable X_h is the state of a message forwarding during a routing process. Specifically, X_h can be a "failure" state or the ID of the node where the message is located after it is forwarded to the h-th hop [2].

[2]Here, we slightly abuse the notation of node id and the state

- The message is forwarded to the destination or dropped finally ("destination state" and "failure state", respectively). These two states are absorbing and the others are transient [3].

In the following, we will use the feature of the Markov chain to compute the average hit-ratio and the average path length from any source to one specific destination node. We assume that in a P2P system, each node has a unique id. In a system with n nodes, the nodes are named starting from 0 to $n-1$. Without loss of generality, we consider node 0 as a destination, and all other nodes $1, 2, \ldots, n-1$ as sources. We define the state i ($i = 0, 1, \ldots, n-1$) as the state when the message is at node i. We denote n as the "failure" state, i.e., if the message cannot be forwarded to any nodes in the system, it will be dropped and virtually put into n. Obviously, $1, 2, \ldots, n-1$ are transient states and $0, n$ are absorbing states.

We define the *transition probability* $p_{i,j} = \Pr[X_h = j | X_{h-1} = i]$. The matrix of transition probabilities $P = (p_{i,j})_{(n+1)\times(n+1)}$ can be written P as

$$P = \begin{pmatrix} 1 & 0 \cdots 0 & 0 \\ U & Q & V \\ 0 & 0 \cdots 0 & 1 \end{pmatrix}, \tag{1}$$

where Q is an $(n-1) \times (n-1)$ matrix that delineates the transition rates between the transient states $1, 2, \ldots, n-1$. U and V are $(n-1) \times 1$ column matrices that delineate the transition rates between the transient states and the absorbing state 0 and n, respectively.

We define hit ratio $a_{i,j}$ to be the probability that a message is successfully forwarded from a transient state i ($i = 1, 2, \ldots, n-1$) to an absorbing state j ($j = 0, n$), and average path length $m_{i,j}$ to be the average number of steps for the corresponding successful forwarding. Computing these two values are nothing but the absorbing probability problem and the mean first passage time problem respectively. Especially, for the state 0, if we define $A = (a_{1,0}, a_{2,0}, \ldots, a_{n-1,0})^{\perp}$ and $M = (m_{1,0}, m_{2,0}, \ldots, m_{n-1,0})^{\perp}$, by the analysis

[3] If message can retry after previous trials fail, we can model it as a discrete Markov chain with only one absorbing state (destination state). The analysis is almost similar except that you must remove state n and replace $p_{i,i}$ with $p_{i,n}$ on computing transition probabilities.

in [21], we have[4]

$$A = (I - Q)^{-1}U, \quad M = ((I - Q)^{-1}A) \div A. \tag{2}$$

By (2), we can obtain the hit ratio and the average path length from transient state i ($i = 1, 2, \ldots, n-1$) to destination state j ($j = 0, n$). Assuming that a source is uniformly randomly chosen, we have,

Theorem 1: The average *hit ratio* $\bar{a}$ and the average hit path length $\bar{m}$ for the algorithm sending message from any source to the destination 0 are, respectively,

$$\bar{a} = \frac{1}{n}(\pi^0 A + 1), \quad \bar{m} = \frac{1}{n}\pi^0 M, \tag{3}$$

where A, M are defined in (2), and $\pi^0 = (1, 1, \ldots, 1)$.

Based on (3), we can compute the hit ratio and the average path lengths to node 0 from any other nodes. Recall that we made no assumption on node 0, and the derivation of the above formulae does not use any particular feature of this node. Hence, the above formulae are applicable to any node i as the destination. If the system is symmetric such that all nodes are equivalent in terms of looking up, the results of formulae are automatically the hit ratio and the average path length of the system. If the system is not symmetric, the overall performance can be obtained based on the above formulae and the specific features of the system.

In the following, we will demonstrate how to use the above approach to analyze the resilience of Structured P2P systems. Due to space limitations, we just discuss the case of node failures. Please refer [22] for a discussion on resilience under attacks.

We can model node failures as follows: at some instant in time, to one node, say node i, its neighbors may be up or down. Departure or failure of nodes will make some items currently in the routing tables invalid. As defined in [17], there are two main kinds of neighbors: *special neighbors* (such as the successor list in Chord and the Leaf Set in Pastry) and *finger neighbors*. To one node, we define τ_1 and τ_2 as the probabilities that a special neighbor and a finger neighbor are in failure state in term of this node at some point of time respectively (Generally $\tau_1 < \tau_2$).

[4]Define $Z = X \div Y$ as $z_{i,j} = x_{i,j}/y_{i,j}$ for any i, j.

The key to apply the Markov-chain based approach is computing the transition probability $p_{i,j}$ from node i to node j for a given structured P2P system. For each node i, we need to compute the individual probability that node i will forward the message directly to other node j. Obviously, if node j is not a neighbor or itself, the probability of forwarding is 0. For all other nodes, the following steps need to be taken to compute the transition probabilities:

- The set of neighbors, including special neighbors and finger neighbors are determined first. Recall that these two types of neighbors have different failure probabilities.
- Then, the routing semantics specific to the given P2P system are followed to determine the probability of forwarding to each neighbor. Different P2P systems have different routing policy. Care is taken to make sure that the routing policy is honored.

In the following, we use CAN as an example to describe how to compute the transition probabilities. Sylvia et.al [5] proposed CAN as a distributed infrastructure that provides hash table like functionality on internet-like scales. It models the participating peers as zones in a d-dimensional toroidal space. Each node is associated with a hyper-cubal region of this key space, and has only special neighbors, which are the nodes associated with the adjoining hybercubes. Routing consists of forwarding to a neighbor that is closer to the key(in the toroidal space). Let us compute the transition probability $p_{i,j}$ from node i to node j:

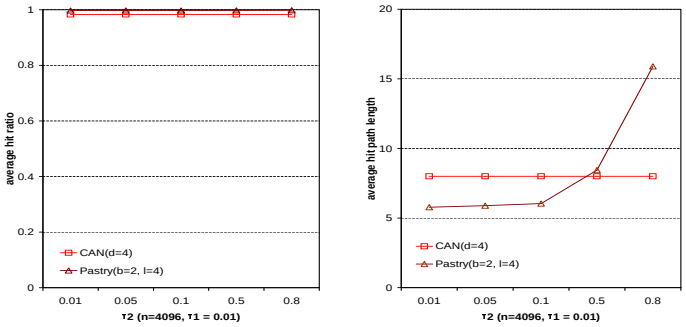
- Given node i, first, let us determine the neighbor set of node i. Recall that the destination is defined as 0. Define $l(i)$ as the lattice distance from i to destination 0 and $l(i,j)$ as the lattice distance from i to j. Define $N(i) = \{i' : l(i') = l(i) - 1 \wedge l(i,i') = 1\}$. $N(i)$ is the neighbor set of node i. All nodes in $N(i)$ are *special neighbors*, and each is operational with probability τ_1. Their lattice distances from node i are same under our assumption.
- Having determined the neighbor set, let us discuss how to follow the routing semantics of CAN to determine the forwarding probability of different neighbors. In CAN system, node i can uniformly choose one of the operational neighbors, and each node in $N(i)$ will be chosen as the next hop with

probability $(1-\tau_1^{|N(i)|})\frac{1}{|N(i)|}$. Now, if all the neighbors are down, the message will be dropped at node i, with probability $p_{i,n}=\tau_1^{N(i)}$.

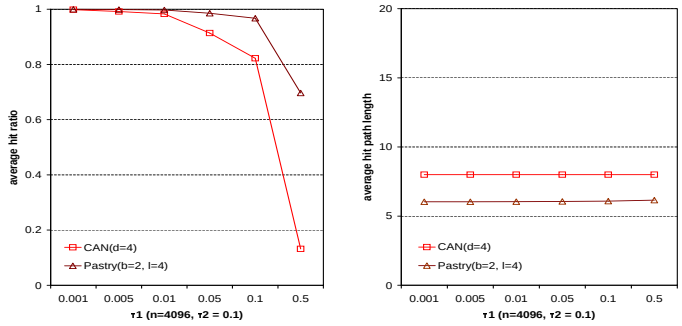
Therefore, for $i=1,2,\ldots,n-1$, we have

$$p_{i,j}=\begin{cases}(1-\tau_1^{|N(i)|})\frac{1}{|N(i)|}, & j\in N(i)\\ \tau_1^{N(i)}, & j=n\\ 0, & \text{otherwise}\end{cases}. \tag{4}$$

We apply the above formulae to compute the average path length and the hit ratio for CAN, Chord, Pastry and Tapestry systems, using MATLAB. We also ran simulation for the systems with the number of nodes ranging from 64 to 4096. Here we only report the data of CAN and Pastry. Note that d is the dimension number for CAN and l is the leaf set size for Pastry. Note that there are no finger neighbors for CAN system but there exist $\mathcal{O}(\log n)$ finger neighbors for Pastry.



(a) finger neighbor



(b) special neighbor

Fig. 1. Performance evaluation of resilience to failure

Fig. 1 illustrate the sensitivity of the average path length and the hit ratio to the failure of the special

neighbors and the finger neighbors for CAN and Pastry, respectively.[5] We have the following observations:

- *Resilience to failures of finger neighbors:* The average path length is very sensitive to the failure of finger neighbors, but the average hit ratio is not, independent of the system model and the number of special neighbors used. Hence, the finger neighbors have significant impact on resilience to failures in terms of the average path length. For example, in the CAN system in Fig. 1(a), both the hit ratio and the average path length remain constant as τ_2 changes for different dimensions, since there are no finger neighbors in CAN; for Pastry, given different sizes of the leaf sets, the average hit ratio is not sensitive to the change of τ_2, unlike the average path length which is very sensitive to such a change.
- *Resilience to failures of special neighbors:* The average hit ratio is very sensitive to the failure of special neighbors, but not the average path length, independent of the system model and the number of special neighbors used. Hence, the special neighbors have significant impact on resilience to failures in terms of the hit ratio. E.g., in Fig. 1(b), as τ_1 increases, the average hit ratio for both CAN and Pastry decreases for different number of special neighbors (*i.e.*, $2d$ in CAN and L in Pastry), but the average path length remains almost constant.

C. Discussions

An important advantage in using a Markov-Chain based approach is its ease of usage in the realm of structured P2P systems. Once the transition probabilities are obtained, it is straightforward to calculate the average path length and hit ratio. Another benefit is the potential ease with which other stochastic theories can be incorporated to analyze other features of P2P systems.

A limitation of the Markov-chain based approach is that, currently it can only be applied to structured P2P systems that are in a stable state. A P2P system may be in a stable state or not. By a *stable* system, we mean a system which obeys the following two criteria: (1) Nodes are uniformly distributed in the system. (2) The number of nodes in the system does not change much over time. It is true that a system

[5]Generally speaking, $\tau_1 < \tau_2$. For the purpose of illustration, we also show the data in the case that $\tau_1 > \tau_2$.

may be in a unstable state due to an intensive attack on the system, or when nodes leave and join the system dramatically. In most cases, a P2P system with DHT's is stable due to the following reasons: (1) To achieve load balancing, the P2P systems attempts to uniformly distribute the nodes in the system. A uniform hash function is used by most systems to achieve this goal. (2) We know that in a P2P system, individual nodes may dynamically leave and join. However, after the initial establishment phase, the number of leaving nodes will match the number of joining nodes in which case the total number of nodes in the system will not change much.

We emphasize here that our work is only a first step towards a generalized analytical approach to analyze resilience of P2P systems. One interesting issue is how to extend our approach to analyse P2P system that are dynamic (absence of a stable state).

IV. Enhancing Resilience of Structured P2P Systems

A. A Survey

This section aims to provide a broad overview of several approaches each of which aim to improve system resilience under failures and attacks. Some approaches exclusively focus on one performance metric to improve while being oblivious to other metrics. Other approaches aim to optimize multiple performance metrics at the same time. Two standard approaches (or methodologies) and some newly emerging approaches that enhance P2P system resilience are described in the following.

1) Redundancy: Enhancing a system with redundancy is critical to the resilience of a P2P system. This includes replication of content, redundant control (or maintenance)/ query messages, availability of multiple routing choices.

The importance of redundancy to improve system resilience is highlighted in [23], [24], [16], [25], [26]. In [24], [26], the problem of efficient storage mechanisms for the data in order to provide high reliabilities is addressed. The authors propose a set of dedicated servers with replication to store content to improve availability. They validate their claim by demonstrating that eventually bandwidth, and not

disk space will be the bottleneck for the P2P applications. Similarly, Sit and Morris [16] highlight the importance of content replication under the threat of freeloading and DDoS attacks. Sufficient replication and avoiding single points of failure is their solution. The authors argue for randomizing the replication both virtually and geographically. Such randomization will also alleviate the effects of coordinated attacks on the system.

The resilience of Tapestry and Pastry is shown to significantly increase under attacks and failures when there are redundancies in control messages and routing paths [23]. Specifically, the authors show that for an N node system by maintaining an additional $logN$ overhead in routing table and an additional log^2N communication overhead (apart from the state currently maintained), data availability significantly increases under failures and malicious nodes. The attack model in this paper typically involves fail stop models, where nodes fail to respond abruptly (node failures). Apart from this, malicious can also generate 'incorrect' messages (routing attacks).

In [25], the authors advocate the use of redundant queries to different replicas to increase data availability in the presence of malicious nodes. The idea is that multiple paths taken by multiple queries increases the likelihood that at least one of the nodes chosen during each hop is non-malicious. Also the presence of multiple routing paths naturally allows more choices for routing. For instance, nodes can route to the nearest node in terms of network delay, thus decreasing latency. Obviously, in the P2P model, multiple paths naturally translate to increases in content availability as one dead end does not mean that the message cannot be routed correctly.

2) Graph Theoretic Approaches: Graph-theoretic approaches to analyze and enhance resilience have been proposed in [15], [27], [18], [19]. The idea here is to introduce 'good' features of graphs to the structures of the P2P systems to improve their resilience. The following discusses graph-theoretic approaches that have been proposed to enhance structured P2P systems resilience.

Topologies like Ring (E.g. Chord [4]) and Hybrid (E.g. Pastry [6]) naturally posses the feature of sequential neighbors (Successor list in Chord and Leaf set in Pastry respectively). Sequential neighbors

are neighbors to which one can route and be sure of making progress towards all destinations. In [15], the importance of adding Sequential neighbors to enhance resilience is shown. Addition of these to the various geometries that can accommodate this feature, significantly increases the hit ratio.

A new P2P structure based on optimal diameter de Bruijn graphs is presented in [18]. de Bruijn graphs are those that are close to achieving the optimal diameter for a given graph. Typically, de Bruijn graphs can achieve a diameter of $D = log_k N$, where N is number of nodes and k is the fixed node degree [20]. This drastic reduction in diameter achievable by de Bruijn graphs is the key motivation of [18]. For a system with $N = 10^6$ nodes, Chord has a node degree and diamter equal to $logN = 20$. On the other hand with a fixed node degree $k = 20$, de Bruijn graphs have diameter of $log_k N = 5$. This implies improved routing efficiency and better resilience properties.

In [27], the authors design a localizer algorithm that attempts to design the P2P system in such a way that; it is reflective of the underlying topology in order to minimize load at the network layer; it maintains connectivity among alive nodes in the system even under the presence of faults and dynamics; the cost of using the overlay is distributed evenly among all participants. In a long term execution of this localizer algorithm, the resultant topology is representative of the classical Erdos and Renyi model [28] that has the properties that this paper addresses apart from the proximity requirement.

Saia et. al. defined a notion of ϵ-dynamically strong fault-tolerance in [29] to describe the robustness of CAN under massive targeted attacks in highly dynamic environments. A CAN network initially containing N peers is ϵ-dynamically strong fault-tolerant if, with high probability, all but an ϵ fraction of the peers in the CAN system can access all but an ϵ fraction of the data items during a period when a certain number of peers polynomial in N are removed by an adversary. They then design a virtual CAN which is ϵ-dynamically strong fault-tolerant. The virtual CAN includes totally N virtual nodes. The virtual nodes and the data items are mapped on the supernodes in a butterfly network of depth $logN - log(logN)$ by a set of hash functions.

An approach based on construction of Random Graphs, where points represent resource identifiers is

presented in [19]. In this graph, the probability of a connection between two nodes depends only on the distance between the nodes. The key principle employed by the authors is that in random graphs, failures result is a smaller random graph that can be used to locate existing resources.

In [30], the small world model [31] is applied to enhance the routing performance of freenet which is an unstructured P2P system. Inspired by this work, in [17] we augment the resilience of a structured P2P system, i.e. CAN with this model. A network is said to exhibit the small-world phenomenon if, roughly speaking, any two nodes in the network are likely to be connected through a short sequence of intermediate nodes. One network construction that gives rise to small-world behavior is one in which each node in the network knows its local neighbors, as well as randomly chosen remote nodes.

3) Miscellaneous Solutions: Apart from the above, other novel approaches using Cryptography and Biological principles have also been proposed to improve resilience.

Sit and Morris [16] discuss the use of authentication and cryptography techniques to enhance system resilience. They propose many attack scenarios. Their approaches to defend against attacks include; assignment of keys to nodes to be done in a verifiable, trusted manner; the design of strong invariant properties that are required to be maintained and verified during dynamics; authenticating the joining process by a trusted authority; keeping track of the node activity by means of signatures and maintaining a report of a node's behavior to check for maliciousness.

In [25], the authors discuss the importance of; having a secure assignment of node identifiers; secure routing table maintenance; and secure message forwarding. For securing the assignment of identifiers to nodes, the authors propose a heavy weight and a light weight solution in the form of using a centralized server to assign identifiers or ask individual nodes to solve crypto puzzles before being permitted into the system. This approach also increases the security in forwarding. Stronger constraints on neighborship and addition of trusted bootstrapping nodes will increase security in routing.

A recent paradigm of computation makes use of biological mechanisms [32], [33] to address problems in computer science. In this framework a biological framework for realization of resilient P2P system

is explored in [34]. Here the system model is distributed computing rather than purely file searching. In this context, they model the system deriving inspiration from the Complex Adaptive Systems (CAS) paradigm. The authors provide preliminary discussions on a very simple approach, the core idea of which is as follows. The resources (computational) are modeled as nests and queries to find out resources are modeled as ants. In a purely random fashion, ants traverse the network to find desirable nests. They may also move computation from an overloaded nest to a non overloaded one. In such a manner, the system is inherently fault tolerant, resilient to random failures of nests and self- organizing. However many implementation details need to be analyzed before such models can be applied for making P2P systems resilient. Nevertheless, biological mechanisms do provide an interesting area to explore in this regard.

B. Enhancing Resilience of CAN and Chord

In the following, we describe our work in enhancing resilience of two structured P2P systems, namely CAN and Chord.

1) CAN-SW: In [17], we apply the small-world model in CAN to form CAN-SW. In the basic CAN-SW system, there are three key issues in the looking-up system: *CAN-SW construction*, *routing mechanism*, and *CAN-SW maintenance*. In CAN-SW, we still follow the greedy routing mechanism. In each step, the current message-holder chooses as the next hop a neighbor (either a local one or a remote one) that is as close to the destination as possible, in the sense of lattice distance. The maintenance mechanisms are almost same as those in CAN system. The only difference is that remote neighbor nodes have to be taken into account in the same manner as local neighbors. In the following, we focus on CAN-SW construction.

In the original design of the CAN system, each node maintains as its set of neighbors, the IP addresses of those nodes that hold coordinate zones adjoining its own zone. In our system, besides local neighbors, each node will maintain an additional remote neighbor. The remote neighbor v of node u is chosen randomly with probability $\alpha_{i,j} = \frac{l^{-d}(i,j)}{\Delta}$, where $\Delta = \sum_{i' \neq i} l^{-d}(i,i')$. In order to choose remote neighbors, we need to know the size of system(*i.e.*, the number of nodes/zones) and the topology of the base CAN. Due to

the dynamics of the construction of CAN, the topology of the base CAN is not regular. For simplicity, we assume that the topology is a d-torus. We can use the volume of individual zones to measure the system size. For example, for node i associated with zone Z_i, we define $[Z_i]$ as its volume. Now we know that the overall system volume is 1. So the system size is estimated to be $n \approx m^d$, where $m^d \leq \frac{1}{[Z_i]} < (m+1)^d$. Once n is known, based on node i, a remote point v can be generated with probability $\alpha_{i,j}$. Using the routing and lookup mechanism, the remote zone where j is located can be found.

Note that the addition of a new node affects only a small number of existing nodes in a very small locality of the coordinate space. The number of neighbors a node maintains depends only on the dimension d of the coordinate space and the number of remote neighbors [6], and is independent of the total number of nodes in the system. Thus, node insertion affects only $\mathcal{O}(d)$ existing nodes, which is important for CAN-SW with huge numbers of nodes.

In [17], we apply the Markov-chain based approach to analyze the resilience of CAN-SW. We first follow the step described before to compute the transition probabilities of CAN-SW and compute the average path length and the hit ratio.

To compute $p_{i,j}$, first, we consider the neighbors, especially the remote neighbors. Although small-world model considers all remote neighbors as being generated initially, at random, we invoke the "Principle of Deferred Decisions" – a common mechanism for analyzing randomized algorithms [14] – and assume that the remote neighbors of a node i are generated only when the message first reaches j. We know that the probability that i choose j as remote neighbor is $\alpha_{i,j} = l^{-d}(i,j)\frac{1}{\Delta}$, where $\Delta = \sum_{i' \neq i} l^{-d}(i,i')$. Define the set of remote neighbors that can be the next hop as $B(i) = \{i' : l(i') \leq l(i) - 1\} \wedge l(i,i') > 1\}$. $N(i) = \{i' : l(i') = l(i) - 1 \wedge l(i,i') = 1\}$ is defined as the set of local neighbors that can be the next hop. Each node $j \in B(i)$ will be chosen as the next hop with probability $(1-\tau_2)\alpha_{i,j}$. If no remote neighbor is available, a local neighbor must be chosen. Therefore, each node $j \in N(i)$ will be chosen as the next hop with probability $(1-\tau_1^{|N(i)|})\frac{1}{|N(i)|}(1-(1-\tau_2)\sum_{i' \in B(i)} \alpha_{i,i'})$. As we know, if all neighbors

[6] So far, we assume it is 1. In the extended version, it may be a constant number larger than 1

are down, the message will be dropped at node i, hence $\tau_1^{|N(i)|}(1-(1-\tau_2)\sum_{i'\in B(i)}\alpha_{i,i'})$. Therefore, for $i=1,2,\ldots,n-1$, we have

$$p_{i,j}=\begin{cases}(1-\tau_2)\alpha_{i,j}, & j\in B(i)\\ (1-\tau_1^{|N(i)|})\frac{1}{|N(i)|} & \\ \quad\cdot(1-(1-\tau_2)\sum_{i'\in B(i)}\alpha_{i,i'}), & j\in N(i)\\ \tau_1^{|N(i)|}(1-(1-\tau_2)\sum_{i'\in B(i)}\alpha_{i,i'}), & j=n\\ 0, & \text{otherwise}\end{cases}. \tag{5}$$

We apply the above formula to compute the average path length and the hit ratio for CAN-SW and compare it with CAN.

TABLE I

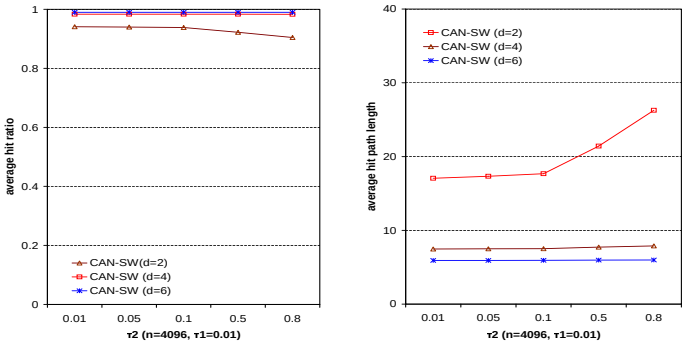
AVERAGE PATH LENGTH FOR CAN AND CAN-SW ($d=2$)

	$n=64$	$n=256$	$n=1024$	$n=4096$
CAN	4	8	16	32
CAN-SW	3.65	6.36	10.74	17.12

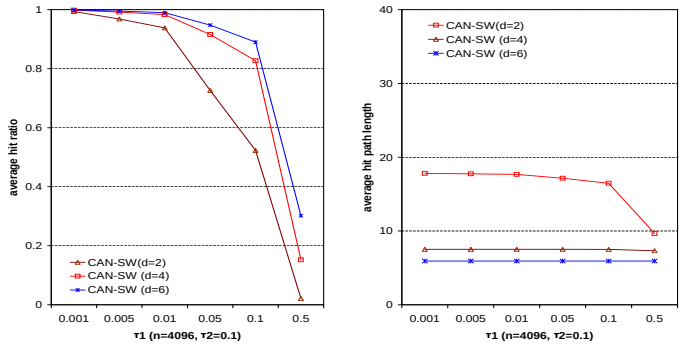
Table I illustrates the comparison between CAN-SW and CAN in terms of the average path length when there is no failure in the system. We can clearly find that CAN-SW outperforms CAN. Particularly, the improvement turns to be more significant as the number of nodes increases. We expect that the impact will increase as the number of remote neighbors increases, however it will cost more maintenance overhead.

Fig. 2 illustrates the sensitivity of the average path length and the hit ratio to the failure probabilities of the local neighbors (the special neighbors) (τ_1) and the remote neighbors (the finger neighbors) (τ_2) of different systems. It confirms the observation in Section III-B that the average path length is sensitive to the remote neighbors, while the hit ratio is sensitive to the local neighbors. Once $\tau_2=1$, CAN-SW becomes CAN. Fig. 2(a) (where τ_1 is fixed, and $\tau_2\to 1$) shows that CAN-SW outperforms CAN both in terms of hit ratio and the average path length.

2) RChord: Chord is susceptible to hostile routing attacks due to its uni-directional routing mechanism. CAN and Pastry support bi-directional routing. Hence even if a query overshoots the destination, a benign



(a) finger (remote) neighbor



(b) special (local) neighbor

Fig. 2. Resilience to failure of remote and local neighbors for CAN-SW

node will be able to correctly route it by backtracking to the destination in the case of CAN and Pastry, but it is not possible in Chord. This prompts us to believe that Chord's resilience can be greatly improved by incorporating bi-directional routing. To enable bi-directional routing, in [35], we propose to add reverse edges to Chord. The new Chord system is called RChord. Two important issues were addressed in: (1) How to add the reverse edges to the routing table? (2) How to do routing with reverse edges?

One simple approach in adding reverse edges is to include the nodes which are the *mirror nodes* for the nodes present in the routing table of a normal Chord node. Obviously, more reverse edges will lead to better performance, however, it would introduce more overhead and redundancy among the edges. We believe that significant performance improvement can be achieved only by adding a few reverse edges. We focus on adding a constant number of reverse edges and propose several approaches to add them. We broadly classify them as deterministic and randomized algorithms.

With deterministic algorithms, the nodes that the reverse edges point to are determined by some system

parameters, for example, the system size. The way the current Chord system adds edges is deterministic. That is, each node has a fixed set of neighbors located at predetermined distances relative to the nodes. We design several deterministic algorithms: (1) The Mirror Algorithm (M): with this algorithm, the reverse edges of a given node are mirrors of the fingers (the clock-wise edges in the original Chord system) of that node in the system. (2) The Uniform Algorithm (U): Assume the number of reverse edges is a constant number, say R. With this algorithm, the R reverse edges are distributed uniformly the space of $\log$ scale. The whole space is logged into R sections; (3) The Local-Remote Combination Algorithm (L-R): This algorithm attempts to follow the idea of cooperation between local edges (L)and remote edges (R). We add some edges close to the current node, while add others at a remote distance. They are chosen alternatively.

With the randomized algorithm, the local reverse neighbors are still selected following the same way as algorithm *L-R*, however, the remote reverse neighbors are selected in random fashion. Specifically, the probability of one node to be selected as the remote reverse neighbor is proportional to the distance between the current node and that node. With this algorithm, the node which is far away from the current node has higher probability to be chosen as a remote node. In this sense, the algorithm is similar to the above deterministic local-remote combination algorithm. However, some nodes which are closer to the current node also have some probability to be selected and so we expect the performance to be different from the deterministic one.

The formal descriptions of the above algorithms are given in Figure 3.

We need to extend Chord's routing algorithm to consider the reverse edges. Chord's routing algorithm is greedy. Upon receiving a request, a peer will pick a node among its neighbors which is closest to the destination in the clock-wise direction. The algorithm is simple and robust. A simple extension is picking the neighbor among the forward and reverse neighbors which is closest to the destination in any one of the two directions. While the algorithm is simple, also compatible with the original algorithm, it is not efficient [35]. The basic reason for that is the asymmetry between the clock-wise edges and the reverse

Input: the system size N ($m = \log N$), the number of reverse edges R, the current node id i

Output: the reverse neighbors r_k ($k = 0, \ldots, R-1$)

Mirror Algorithm:

$r_k = (i - 2^k) \bmod 2^m$, where $k = 0, \ldots, R-1$.

Uniform Algorithm

$r_k = (i - 2^k) \bmod 2^m$, where $k = \lfloor \frac{(p+1)m}{R+1} \rfloor$, $p = 0, \ldots, R-1$.

Local-Remote Combination Algorithm

$r_k = (i - 2^k) \bmod 2^m$, where $k = (m - 1 - \lfloor \frac{p}{2} \rfloor)(p \bmod 2) + \lfloor \frac{p}{2} \rfloor (1 - p \bmod 2)$, $p = 0, \ldots, R-1$.

Local-Remote Random Algorithm

$r_k = (i - 2^k) \bmod 2^m$, when k is even and less than R.

The probability of a node j to be r_k (where k is odd and less than R) is,

$Prob(j) = (i - j) \bmod 2^m / \sum_{q=1}^{n/2-1} q \bmod 2^m$.

Fig. 3. Algorithms for Adding Reverse Edges for the Chord System

edges. We conclude that our extension should consider the asymmetry between the forward edges and the reverse edges. Based on the above consideration, we design a routing algorithm which (1) is compatible to the original Chord routing algorithm, (2) considers the cooperation between the forward and reverse edges, and the difference in the routing capacity of the forward and reverse edges.

In [35], our analysis results show that the RChord system is much more resilient to routing attacks than the original Chord system. Due to space limitations, we do not report data here.

C. Discussions

The two approaches we have proposed to enhance resilience basically are inspired from the ideas of redundancy and the richness of graph theory.

The remote edges in the CAN-SW system and the reverse edges in the RChord system (both point to neighbors added to the routing table at a node) are fundamentally added to the existing neighbor list at

nodes in the system. That is, both increase the redundancy in terms of routing choices at each node when forwarding a message. Under node failures or attacks, the *new* neighbors help in successfully routing messages thus improving the system resilience in terms of Hit ratio.

However, while adding these new neighbors, we apply ideas in graph theory with the objective of significantly improving system resilience. In CAN-SW, the remote edges are added randomly following the small world model, while in RChord, a randomized remote edge adding algorithm was designed. Randomly selected reverse edges will make it difficult even for a highly destructive attack to cause extensive damage.

There are several directions to extend our work. For CAN-SW, the remote neighbor(s) can be added considering several factors: (1) Load balancing: if each node constructs multiple remote neighbors independently, the performance would improve due to better load balancing in the routing. However, this may increase the overhead in system maintenance. (2) Considering the issue of popularity and importance of a node, we can adjust the number of remote neighbors for each node by assigning relative weights to it.

It will be an interesting exercise to theoretically prove the features of our reverse edge adding algorithms in our RChord system and hence derive an algorithm to add these edges that will give optimum performance.

V. Final Remarks

In this paper, we provided a detailed summary on the resilience of structured P2P systems under the presence of node dynamics and attacks. We first provided a brief introduction to the design features and performance metrics of structured P2P systems followed by threats associated with structured P2P systems.

We then described approaches to analyze the resilience of structured P2P systems. Our discussions on analysis first provided with a survey of related work. We then proposed our analytical approach to analyze resilience of structured P2P systems under failures. The approach is Markov-chain based, and can be applied to systems with relatively stable size and uniform distribution of nodes.

Finally, we discussed several enhancements that have been proposed to the design features of structured P2P systems to make them more resilient. We then discussed our approaches to enhance the design features of two popular structured P2P systems, namely CAN and Chord. Both our enhancements employed redundancy and graph-theoretic approaches for the enhancements. Specifically, the new edges (or the neighbors) that were added to our CAN-SW and RChord system increased redundancy which translates to more routing choices. This again translates to improved resilience under failures or attacks. For adding the edges we employed principles of Small world and random graphs such that even under highly destructive attacks, the impacts are minimized.

REFERENCES

[1] "Gnutella," http://www.gnutellaforums.com/.

[2] "Kazaa," http://www.kazaa.com.

[3] "Freenet," http://freenet.sourceforge.net/.

[4] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2003 ACM Special Interest Group on Data Communication (SIGCOMM)*, August 2001.

[5] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, "A scalable content-addressable network," in *Proceedings of the 2003 ACM Special Interest Group on Data Communication (SIGCOMM)*, Auguest 2001.

[6] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001.

[7] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," in *IEEE Journal on Selected Areas in Communications*, January 2004.

[8] R. Rivest, "The md5 message-digest algorithm," Internet RFC, April 1992.

[9] P. Jones, "Us secure hash algorithm 1 (sha1)," Internet RFC, September 2001.

[10] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in *Proceedings of ACM SPAA. ACM*, June 1997.

[11] Dahlia Malkhi, Moni Naor, and David Ratajczak, "Viceroy: A scalable and dynamic emulation of the butterfly," in *Proceedings of the Twenty-First ACM Symposium on Principles of Distributed Computing (PODC)*, July 2002.

[12] Petar Maymounkov and David Mazires, "Kademlia: A peer-to-peer information system based on the xor metric," in *Proc. of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[13] John R. Douceur, "The sybil attack," in *Proceedings of First International Workshop on Peer-to-Peer Systems (IPTPS) Workshop*, March 2002.

[14] Dennis Heimbigner, "Adapting publish/subscribe middleware to achieve gnutella-like functionality," in *Proceedings of the 2001 ACM symposium on Applied computing*, 2001.

[15] Krishna P. Gummadi, Ramakrishna Gummadi, Steven D. Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica, "The impact of dht routing geometry on resilience and proximity," in *Proceedings of the ACM SIGCOMM Symposium on Network Architectures and Protocols*, August 2003.

[16] Emil Sit and Robert Morris, "Security considerations for peer-to-peer distributed hash tables," in *Proceedings of First International Workshop on Peer-to-Peer Systems (IPTPS) Workshop*, March 2002.

[17] Shengquan Wang, Dong Xuan, and Wei Zhao, "On resilience of structured peer-to-peer systems," in *IEEE Global Telecommunications Conference (GLOBECOM)*, December 2003.

[18] Dmitri Loguinov, Anuj Kumar, Vivek Rai, and Sai Ganesh, "Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience," in *Proceedings of the 2003 ACM Special Interest Group on Data Communication (SIGCOMM)*, August 2003.

[19] James Aspnes Zoe Diamadi and Gauri Shah, "Fault-tolerant routing in peer-to-peer systems," in *ACM Symposium on Principles of Distributed Computing*, July 2003.

[20] M. Imase and M. Itoh, "Design to minimize diameter on building-block network," *IEEE Trans. on Computers*, vol. 30, 1981.

[21] A. Ravindran, Don T. Phillips, and James J. Solberg, *Operations Research: Principles and Practice, 2nd Edition*, John Wiley and Sons, January 1987.

[22] Shengquan Wang and Dong Xuan, "A markov-chain based approach to resilience of structured p2p systems under failures and attacks," Technical Report, Dept. of Computer Science and Information, The Ohio-State University, February 2003.

[23] Kirsten Hildrum and John Kubiatowicz, "Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks," in *Proceedings of 17th International Symposium on Distributed Computing*, October 2003.

[24] Rodrigo Rodrigues, "An agenda for robust peer-to-peer storage," in *First Infrastructure for Resilient Internet Systems (IRIS) Student Workshop*, August 2003.

[25] Micuel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[26] Charles Blake and Rodrigo Rodrigues, "High availability, scalable storage, dynamic peer networks: Pick two," in *Workshop on Hot Topics in Operating Systems (HotOS)*, May 2003.

[27] Laurent Massouli, Anne-Marie Kermarrec, and Ayalvadi J. Ganesh, "Network awareness and failure resilience in self-organising overlays networks," October 2003.

[28] Hongsuda Tangmunarunkit, Ramesh Govindan, Sugih Jamin, Scott Shenker, and Walter Willinger, "Network topologies, power laws, and hierarchy," Tech. Rep. TR01-746, Technical Report, University of Southern California, June 2001.

[29] Jared Saia, Amos Fiat, Steve Gribble, Anna R. Karlin, and Stefan Saroiu, "Dynamically fault-tolerant content addressable networks," in *Proceedings of First International Workshop on Peer-to-Peer Systems (IPTPS) Workshop*, March 2002.

[30] Hui Zhang, Ashish Goel, and Ramesh Govindan, "Using the small-world model to improve freenet performance," in *Proceedings of the IEEE INFOCOM*, June 2002.

[31] Jon Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[32] Jeffrey O. Kephart, "A biologically inspired immune system for computers," in *Proceedings of the Fourth International Workshop on Synthesis and Simulatoin of Living Systems*, 1994.

[33] Michael Wang and Tatsuya Suda, "The bio-networking architecture: A biologically inspired approach to the design of scalable, adaptive, and survivable/available network applications," in *Proceedings of the 1st IEEE Symposium on Applications and the Internet (SAINT)*, January 2001.

[34] Alberto Montresor, Heing Meling, and Ozalp Babaoglu, "Towards self-organizing, self-repairing and resilient large-scale distributed systems," in *Proceedings of the 1st International Workshop on Future Directions in Distributed Computing*, June 2002.

[35] Dong Xuan, Sriram Chellappan, and Muralidhar Krishnamoorthy, "Rchord: An enhanced chord system resilient to routing attacks," in *Proceedings of IEEE International Conference on Computer Networks and Mobile Computing (ICCNMC)*, October 2003.